ML-based Intrusion Detection on UNSW-NB15

Niaz Morshed
Doctor of Information Technology,
College of Graduate and Professional Studies,
Trine University, USA

ABSTRACT
Machine learning (ML)-based network intrusion detection systems (NIDS) are typically compared by accuracy on public benchmarks, but such scores reveal little about how a detector performs when traffic, attacks, or the monitoring pipeline change. This paper presents a leakage-aware and resilience-oriented evaluation of four detectors (logistic regression, random forest, XGBoost and a multilayer perceptron) on the official UNSW-NB15 partition. A data audit reveals that 42.2% of training records are exact duplicates and 10.4% of test flows also occur in training. Random forest and XGBoost reach the highest F1 scores (89.43% and 89.37%), yet at the default decision threshold they generate alarms for approximately 27% of normal traffic, and thresholds calibrated to a 1% false-positive rate on validation data yield 2.4% to 4.9% on the test set. A single rule on the source time-to-live (TTL) value reaches 76.63% accuracy, and TTL is 2.8 times as influential as the next-ranked feature. When each attack family is removed from training, detection at a 1% false-alarm budget falls from 90.1% to 75.1% on average; for Generic attacks it falls from 99.8% to 51.3%, and to 23.3% without TTL features. Under simulated telemetry loss, tree ensembles degrade gracefully while the linear model loses up to 36.5 F1 points. These findings support reporting operating-point, novelty and degradation metrics alongside headline accuracy.

Keywords
Intrusion detection, UNSW-NB15, data leakage, shortcut learning, cyber resilience, zero-day attacks, explainable AI

1. INTRODUCTION
Power grids, water utilities, transport, healthcare, and financial services now depend on networked information and control systems, so a failure of network security can quickly become a failure of the service itself. Network intrusion detection systems are one of the primary defensive layers in this setting, and machine learning has been studied for many years as a way to detect attacks that hand-written signatures miss [4, 6]. Results on public benchmark datasets provide the primary evidence used to compare such detectors [4, 25].

Benchmark scores, however, answer a narrow question. Sommer and Paxson argued that intrusion detection differs from other ML applications because errors are expensive, attacks are diverse and continually evolving, and realistic evaluation data are scarce [29]. Axelsson showed that because attacks occur infrequently, even a small false-alarm rate can make a detector impractical [3]. Later studies found heavy record redundancy in KDD Cup 99 [30], labeling and feature-extraction errors in CICIDS2017 [10], and recurring pitfalls in security ML such as spurious correlations, data snooping and inappropriate performance measures [2]. When a model exploits a dataset-collection artifact rather than the behaviour of interest, it exhibits what Geirhos et al. call shortcut learning: good benchmark scores that do not survive a change in conditions [11].

For digital infrastructure, resilience is a critical property. NIST defines cyber resiliency as the ability to anticipate, withstand, recover from and adapt to adverse conditions, stresses, attacks or compromises on systems that use or are enabled by cyber resources [26], and Linkov and Kott distinguish resilience, which concerns recovery and adaptation after a disruption, from the narrower aim of preventing it [15]. For an intrusion detector, resilience has at least three dimensions: it should keep a usable false-alarm rate when traffic shifts, it should still flag attack types it was not trained on, and it should degrade gracefully when part of the monitoring pipeline fails. Headline accuracy captures none of these properties.

This paper evaluates ML intrusion detection on UNSW-NB15 [19, 20] from this perspective. Six research questions guide the study: (RQ1) how clean the official train/test partition is; (RQ2) how four common detectors compare, including on test flows never seen in training and at a fixed false-alarm budget; (RQ3) which attack families are confused; (RQ4) whether the models rely on a testbed artifact; (RQ5) how well attack families absent from training are detected; and (RQ6) how detection degrades under telemetry loss. The contributions are:

(1) A data audit of the official UNSW-NB15 partition that quantifies duplication (42.2% of training rows), train/test overlap and label conflicts, and documents a file-naming inconsistency that can silently exchange the training and test sets.

(2) A benchmark of logistic regression, random forest, XGBoost and a multilayer perceptron over 3 seeds with exact McNemar tests, reported at the default threshold and at a 1% false-alarm budget.

(3) Evidence of a time-to-live (TTL) shortcut from a single-feature baseline, feature ablation and SHAP attributions.

(4) A leave-one-family-out zero-day proxy, run with and without the shortcut features, which identifies the sources of apparent generalization.

(5) A telemetry-degradation analysis covering random loss of feature values and the outage of whole feature-extraction modules.

2. RELATED WORK
2.1 Machine learning for intrusion detection
Buczak and Guven surveyed data mining and ML methods for cyber intrusion detection [6], and Ring et al. reviewed the network-based datasets used to evaluate them [25]. A recent survey in this journal examined both the algorithms and the evaluation practices of ML-based intrusion detection [4]. UNSW-NB15 was created because older benchmarks no longer reflected modern traffic and low-footprint attacks [19], and its statistical properties were compared with KDD99 in [20]. Its creators also used it to build statistical anomaly detectors based on finite Dirichlet mixture models [21] and geometric area analysis [22], and a NetFlow version was later released together with other benchmarks [27]. The dataset is now widely used: Kasongo and Sun studied feature selection with several classifiers on it [12], and Vinayakumar et al. evaluated deep neural networks on it

alongside other datasets [32]. Sarhan et al. proposed a common NetFlow-based feature set so that detectors can be compared across datasets [28]. Machine learning is increasingly promoted as the backbone of proactive supply chain risk management, and a large number of recent studies report near-perfect accuracy for disruption prediction on small public datasets [40].

Recent studies continue to compare ensemble and deep architectures on security datasets. FORTIS uses a hybrid ensemble for industrial cybersecurity and intrusion detection [35], while SecureEnsembleNet applies ensemble learning to cyber-threat detection [36]. Badjie et al. compare ensemble and graph neural network approaches for illicit Bitcoin transaction detection under temporal drift [34]. A cross-domain comparison of deep learning and conventional ML for RNA-seq cancer-type identification provides another example of benchmark-based model comparison [37].

2.2 Pitfalls in evaluation

Several studies have questioned how ML detectors are evaluated. Beyond the issues raised in [29] and [3], Arp et al. found that pitfalls such as sampling bias, spurious correlations and lab-only evaluation are common in security research [2]. Pendlebury et al. showed that ignoring spatial and temporal constraints inflates malware classification results [24], and Kaufman et al. formalized leakage, in which information that would not be available at prediction time enters the learning process [13]. Dataset-level problems have been documented for KDD Cup 99 [30] and CICIDS2017 [10]. Apruzzese et al. argued that research rarely lets practitioners judge the real-world value of ML-based NIDS and proposed a pragmatic assessment that accounts for deployment factors [1].

Leakage-aware evaluation is also emerging in adjacent security tasks. Sharif et al. identified label leakage in a public Ethereum fraud dataset, removed the leaking fields, and used SHAP to explain the corrected detector [33]. This result parallels the present paper's concern that high scores can reflect artifacts rather than durable behavioral signals.

2.3 Resilience and unknown attacks

The ability to detect unseen attacks is often claimed but rarely assessed directly. Kus et al. revisited ML-based industrial intrusion detection and found that near-perfect reported scores came from testing on attack types already seen in training; on held-out attack types detection dropped sharply [14]. This study applies the same held-out approach to UNSW-NB15 and combines it with a shortcut analysis, an operating-point analysis and a telemetry-failure analysis. We are not aware of any UNSW-NB15 study that reports these four views together.

Robustness to incomplete inputs has also been studied outside NIDS. Nandy et al. evaluated digital-twin estimation under random and burst sensor-data loss [38], while Bouraima et al. tested predictive analytics under controlled missing-data regimes [39]. Their settings differ from network-flow detection, but both support reporting how model performance changes as telemetry becomes incomplete.

3. DATASET AND DATA AUDIT

3.1 UNSW-NB15

UNSW-NB15 was generated in the Cyber Range Lab of the Australian Centre for Cyber Security with the IXIA PerfectStorm tool, which produced a hybrid of real normal traffic and synthetic attack traffic [19]. Flows are described by 49 features derived with Argus, Bro-IDS and twelve additional algorithms, and attacks belong to nine families: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode and Worms [19]. The creators distribute the dataset, including a train/test partition of 175,341 and 82,332 records, through the official UNSW-NB15 page [31], from which the partition files used here were obtained. The partition contains 42 features because identifiers such as IP addresses, ports and timestamps are removed. In the copy of the partition used here, the file named UNSW_NB15_training-set.csv holds 82,332 records and UNSW_NB15_testing-set.csv holds 175,341, reversing the published description. We assign the larger file to training to match that description. A pipeline that loads the files by name would therefore train on the smaller set without warning.

3.2 Audit results

Table 1 summarises the class distribution. The audit identified four issues. First, 74,072 training rows (42.2%) are exact duplicates of another row, including the label; removing them leaves 101,269 rows. Duplication is uneven across classes: 90% of the Generic training rows are duplicates (40,000 to 3,924), and DoS shrinks from 12,264 to 3,004. As a result the share of normal traffic is 31.9% in the raw training file, 51.2% after de-duplication and 44.9% in the test file. Second, 229 distinct feature vectors appear in training with both labels, which limits the accuracy achievable on those flows. Third, 8,541 test flows (10.4%) have a feature vector identical to a training flow; we therefore also report results on the remaining 73,791 unseen test flows. Fourth, the connection states ACC and CLO occur in the test file but never in training. The test file (which also contains 28,380 duplicates) is left unchanged to preserve comparability with prior work.

Table 1: Class distribution of the UNSW-NB15 partition before and after de-duplication

Class	Train (raw)	Train (dedup.)	Test	Test (unseen)
Normal	56,000	51,890	37,000	35,528
Analysis	2,000	542	677	552
Backdoor	1,746	513	583	579
DoS	12,264	3,004	4,089	2,085
Exploits	33,393	18,721	11,132	9,330
Fuzzers	18,184	15,043	6,062	5,801
Generic	40,000	3,924	18,871	17,149
Reconnaissance	10,491	6,422	3,496	2,358
Shellcode	1,133	1,089	378	365
Worms	130	121	44	44
Total	175,341	101,269	82,332	73,791

4. METHODOLOGY

4.1 Preprocessing

The 42 features comprise 39 numeric features and three categorical ones (proto, service and state). Numeric features are transformed with a signed $\log(1 + |x|)$ to compress heavy tails and then standardised. Categorical features are one-hot encoded, with levels observed fewer than 50 times grouped into one "infrequent" level that also absorbs unseen test categories. All transformations are fitted exclusively on the training data. From the de-duplicated training set, a stratified 10% validation split is held out; it is used only to set the alert threshold described in Section 4.3, never for training or model selection.

4.2 Detectors

Four detectors commonly used in the NIDS literature were trained with hyperparameters fixed before examining any test results: logistic regression (LR, L2 penalty, L-BFGS solver); a random forest (RF) [5] with 200 trees; XGBoost [7] with 400 trees, maximum depth 8, learning rate 0.1, and row and column

subsampling of 0.8; and a multilayer perceptron (MLP) with hidden layers of 128 and 64 ReLU units, Adam, batch size 512 and early stopping. Implementations come from scikit-learn [23] and the XGBoost library. Each configuration was trained with 3 random seeds and results are reported as mean and standard deviation.

4.3 Metrics and operating point

Binary results are reported as accuracy, precision, recall, F1, the Matthews correlation coefficient (MCC), which remains informative under class imbalance [8], and the area under the ROC curve. F1 is also computed on the unseen test subset. Because an operator configures a detector by its false-alarm budget rather than by a probability of 0.5, each model also receives an operational threshold: the 99th percentile of its scores on normal validation flows, to target a 1% false-positive rate (FPR). We report the true-positive rate (TPR@op) and the FPR (FPR@op) that this threshold produces on the test set. Pairs of models are compared with the exact McNemar test on their test predictions [9, 18]. Training time and per-flow inference latency were measured on a two-core CPU.

4.4 Shortcut analysis

Three analyses assess whether the detectors rely on the time-to-live (TTL) fields, which depend on the testbed's host configuration rather than on attack behaviour: a depth-3 decision tree trained on sttl alone and on the three TTL-derived features (sttl, dttl, ct_state_ttl); XGBoost and RF retrained without these features; and TreeSHAP attributions [16, 17] for XGBoost on 3,000 random test flows. For one-hot encoded features, SHAP values are summed over the encoded columns before the absolute value is taken, which keeps the attribution additive.

4.5 Zero-day proxy

For each of the nine attack families, XGBoost and its preprocessor are retrained with that family removed from training, and its threshold is recalibrated on validation normal flows to 1% FPR. The detection rate on the held-out family's test flows ("unseen") is compared with that of the model trained on all families ("seen"). The experiment is repeated without the TTL features.

4.6 Telemetry degradation

Trained models are evaluated on degraded inputs without retraining. In the random-loss setting each feature value of each test flow is lost with probability p, for p from 0% to 50%, and replaced by the training median (numeric) or mode (categorical), as might be done by a collector; each level is repeated three times. In the module-outage setting a whole feature group from the UNSW-NB15 taxonomy [19] is lost at once: Basic (15 features, including proto, service, state and rate), Content (8), Time (7), General purpose (5) and Connection (7).

All experiments use Python 3.13.16, scikit-learn 1.9.1 and XGBoost 3.4.1; the complete pipeline runs in about 7 minutes on two CPU cores.

5. RESULTS AND DISCUSSION

5.1 Detection performance

Table 2 reports binary detection. RF has the highest mean F1 (89.43%) and MCC (75.54%), followed closely by XGBoost (89.37% and 75.34%). The McNemar test finds no significant difference between the two: of the flows on which they disagree, 1,601 favour one model and 1,604 the other ($p = 0.97$), while both are significantly better than MLP and LR ($p < 10^{-87}$). All detectors have recall above 98% but precision of only 78% to 82%, so most of their errors are false alarms. On the unseen test subset F1 drops by only 1.2 to 1.6 points, suggesting that memorised train/test overlap is not the primary source of these scores. Figure 1 shows the ROC curves on a logarithmic FPR axis, the region most relevant to operational deployment. XGBoost and the MLP detect the most attacks at false-positive rates below 1%, while LR falls far behind below 0.3%.

Table 2: Binary detection on the test set (%, mean ± std over 3 seeds; best in bold)

Metric	LR	RF	XGBoost	MLP
Accuracy	83.66 ± 0.00	**87.17 ± 0.03**	87.13 ± 0.10	86.18 ± 0.56
Precision	77.93 ± 0.00	81.83 ± 0.03	**81.96 ± 0.11**	80.90 ± 1.09
Recall	98.11 ± 0.00	**98.59 ± 0.02**	98.24 ± 0.04	98.08 ± 0.83
F1	86.87 ± 0.00	**89.43 ± 0.03**	89.37 ± 0.08	88.66 ± 0.33
MCC	69.11 ± 0.00	**75.54 ± 0.07**	75.34 ± 0.20	73.61 ± 0.71
ROC-AUC	96.49 ± 0.00	98.05 ± 0.01	**98.25 ± 0.03**	98.01 ± 0.17
F1 (unseen)	85.26 ± 0.00	88.15 ± 0.03	**88.16 ± 0.09**	87.31 ± 0.31

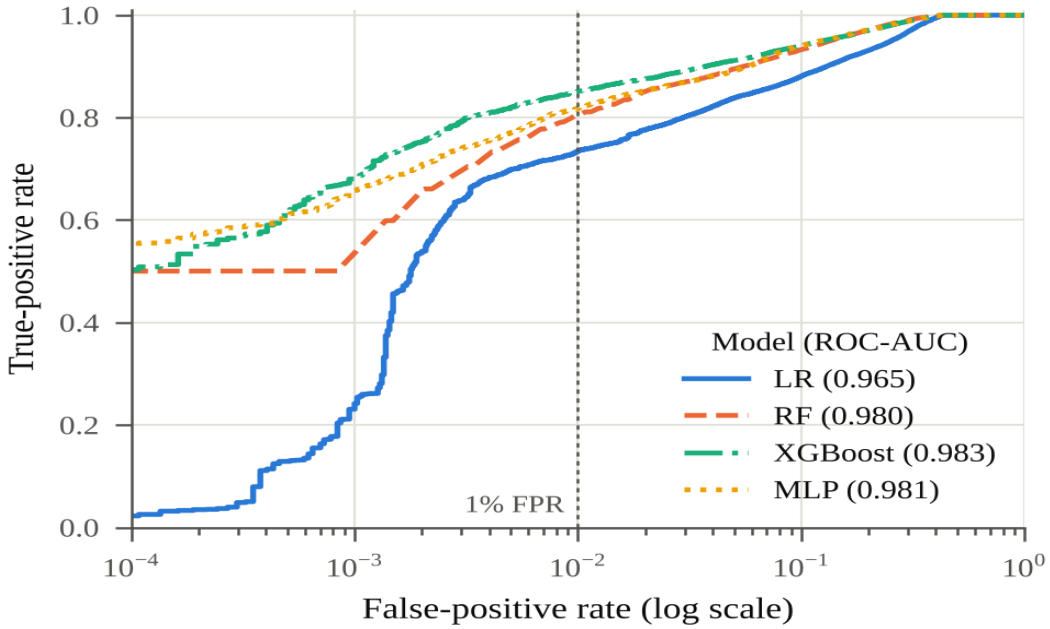


Figure 1: ROC curves (seed 0) with a logarithmic false-positive axis; the dotted line marks 1% FPR

5.2 Operating point and false alarms

Table 3 presents the results from an operational perspective. At the default threshold of 0.5 every detector flags 26.5% to 34.0% of normal test flows as attacks, a rate that would be difficult for a security team to triage. With the threshold set for 1% FPR on validation data, the test FPR is 2.4% to 4.9%, 2.4 to 4.9 times the target. The validation split comes from the training distribution, so this gap measures the distribution shift between the two partitions, and such a shift may also arise in deployment. At this operating point XGBoost detects 90.9% of attacks at 4.9% FPR, and RF detects 87.1% at 3.1%. Following Axelsson's base-rate argument [3], even 3.1% means about 31,000 false alerts for every million benign flows. Computational cost is relatively low: training takes 3.8 to 24.0 seconds and inference 0.17 to 8.14 µs per flow.

Table 3: Operational view: FPR at threshold 0.5, TPR and FPR at the 1% validation-FPR threshold (%), training time and latency

Model	FPR@0.5	TPR@op	FPR@op	Train (s)	µs/flow
LR	34.0 ± 0.0	78.6 ± 0.0	**2.4 ± 0.0**	**3.8**	**0.17**
RF	26.8 ± 0.0	87.1 ± 0.3	3.1 ± 0.3	15.3	8.14
XGBoost	**26.5 ± 0.2**	**90.9 ± 0.2**	4.9 ± 0.2	4.9	4.48
MLP	28.4 ± 2.3	89.4 ± 0.9	4.5 ± 0.3	24.0	1.11

5.3 Attack family identification

Table 4 reports multiclass results for XGBoost trained without (U) and with (B) class-balanced sample weights. Macro-F1 is low in both cases (51.5% and 52.1%). Generic (F1 98.5%) and Reconnaissance (84.7%) are identified reliably, but Analysis (2.7%), Backdoor (7.2%) and DoS (27.4%) are not. The confusion matrix in Figure 2 explains why: these three families are predicted as one another for most of their flows, which suggests that the available flow features do not separate them. Class balancing raises the recall of rare families (Worms from 38.6% to 72.7%, Shellcode from 77.8% to 91.3%) at the cost of normal-traffic recall (75.4% to 62.8%). The matrix also shows that the largest error by flow count is normal traffic predicted as Fuzzers, which links to the false alarms in Section 5.2.

Table 4: Per-family recall and F1 (%) of XGBoost, unweighted (U) and class-balanced (B)

Class	Support	Rec. (U)	F1 (U)	Rec. (B)	F1 (B)
Normal	37,000	75.4	84.6	62.8	76.7
Analysis	677	5.2	2.7	7.5	2.6
Backdoor	583	25.6	7.2	22.8	6.1
DoS	4,089	24.7	27.4	30.1	30.0
Exploits	11,132	68.4	70.9	63.1	69.7
Fuzzers	6,062	58.5	39.4	66.1	37.2
Generic	18,871	97.2	98.5	97.4	98.5
Reconnaissance	3,496	80.8	84.7	83.4	83.3
Shellcode	378	77.8	55.4	91.3	45.0
Worms	44	38.6	44.7	72.7	71.9
Macro avg.			51.5		52.1

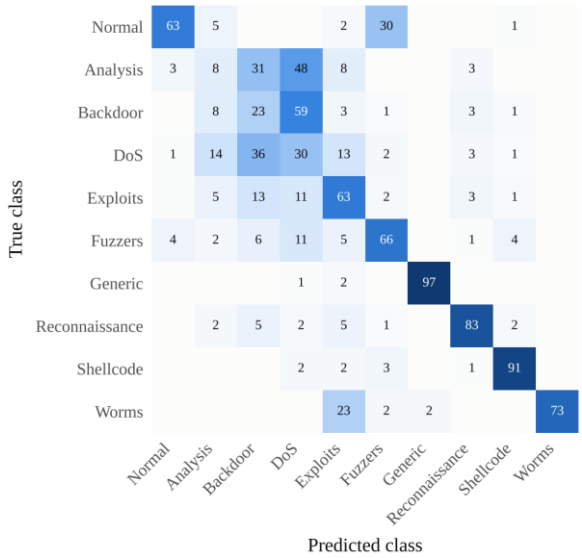


Figure 2: Row-normalised confusion matrix (%) of the balanced XGBoost model; values below 0.5% are hidden

5.4 Shortcut learning

Source TTL separates the classes more strongly than would be expected of a testbed-dependent property. In the raw training file, 39,455 of the 56,000 normal flows have sttl = 31, a value that never occurs in attack flows, while the two most common attack values, 254 and 62, belong mostly to attack flows (78% and 87% of de-duplicated training flows with those values). Table 5 shows that a depth-3 tree on sttl alone reaches 76.63% accuracy and 82.47% F1, and adding dttl and ct_state_ttl raises this to 80.57% and 84.88%. SHAP attributions in Figure 3 rank sttl first with a mean absolute value of 2.59, 2.8 times that of the second feature (service, 0.94); ct_state_ttl is third. Yet removing the three TTL features changes F1 only slightly (XGBoost 89.38% to 89.02%, RF 89.40% to 89.27%). The models therefore rely heavily on TTL, but other features carry largely redundant information about the same split, meaning that in-distribution scores alone cannot expose the shortcut. Section 5.5 shows where it matters. Training with the duplicated rows gives F1 89.26% against 89.38% without them, indicating that de-duplication incurs no accuracy penalty while removing 42.2% of the training data.

Table 5: Shortcut-learning and data-quality ablations (%, seed 0; ref. = all features, de-duplicated training; dupl. = trained with duplicate rows)

Configuration	Acc.	F1	MCC	ROC-AUC	F1 (unseen)
XGBoost (ref.)	87.14	89.38	75.37	98.27	88.17
XGBoost + dupl.	87.01	89.26	75.08	98.23	88.03
XGBoost, no TTL	86.74	89.02	74.44	98.03	87.74
RF (ref.)	87.13	89.40	75.47	98.04	88.12
RF, no TTL	86.97	89.27	75.14	97.87	87.96
Tree, sttl only	76.63	82.47	57.94	72.04	80.49
Tree, TTL only	80.57	84.88	64.19	78.61	83.23

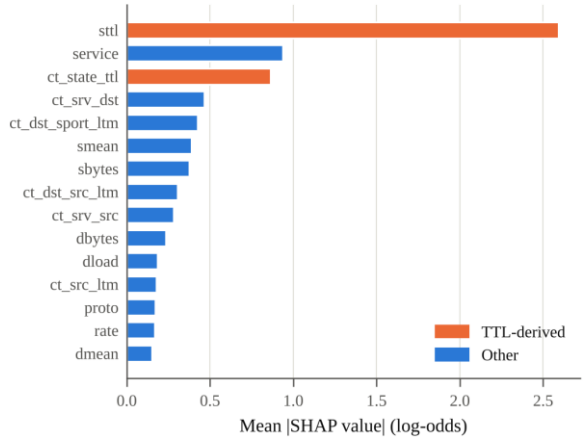


Figure 3: Mean absolute SHAP value of the 15 most influential features for XGBoost; TTL-derived features are highlighted

5.5 Detection of unseen attack families

Table 6 and Figure 4 present the zero-day proxy results at the 1% validation-FPR threshold. On average, a family is detected at 90.1% when it is present in training and at 75.1% when it is held out. This average conceals two distinct patterns. Analysis, Backdoor, DoS and Worms lose at most 2.3 points; given the confusion among Analysis, Backdoor and DoS, a held-out member of this group is still covered by its near neighbours in training. Generic (99.8% to 51.3%) and Reconnaissance (97.9% to 47.7%) lose about half of their detections, and Exploits, Shellcode and Fuzzers lose 10 to 12 points. Without the TTL features, mean unseen detection falls further to 70.9%, and Generic falls to 23.3%: more than half of what looked like generalization to unseen Generic attacks came from the TTL shortcut.

The operating point is as important as the feature set. At the default threshold of 0.5 the same experiment would report 91.9% unseen-family detection against 98.3% seen, a substantially more optimistic picture, but at that threshold about 26% of normal flows raise alarms. The held-out experiments also help identify the source of false alarms: when Fuzzers are excluded from training, the FPR at threshold 0.5 on normal test traffic falls from 26.5% to 5.9%. Most false alarms are therefore normal flows that resemble Fuzzers, consistent with Figure 2.

Table 6: Leave-one-family-out detection rate (%) of XGBoost at a 1% validation-FPR threshold

		All features			Without TTL		
Family	**Flows**	**Seen**	**Unseen**	**Gap**	**Seen**	**Unseen**	**Gap**
Analysis	677	92.0	91.6	0.4	91.7	91.4	0.3
Backdoor	583	98.6	98.5	0.2	99.0	97.9	1.0
DoS	4,089	96.7	95.7	1.0	96.3	95.0	1.3
Exploits	11,132	96.0	84.2	11.8	95.5	84.1	11.4
Fuzzers	6,062	46.1	35.3	10.8	46.4	32.6	13.8
Generic	18,871	99.8	51.3	48.5	99.7	23.3	76.4
Reconnaissance	3,496	97.9	47.7	50.2	97.7	45.5	52.2
Shellcode	378	88.4	78.3	10.1	88.6	75.1	13.5
Worms	44	95.5	93.2	2.3	97.7	93.2	4.5
Mean		90.1	75.1	15.0	90.3	70.9	19.4

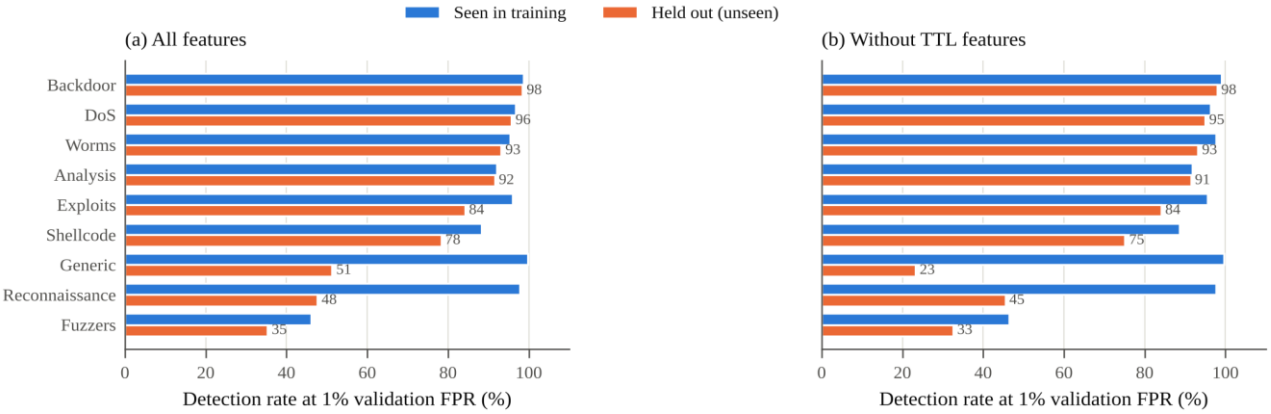


Figure 4: Detection rate at a 1% validation-FPR threshold for each attack family when present in training (seen) and when held out (unseen), (a) with all features and (b) without TTL features

5.6 Robustness to telemetry loss

Figure 5 shows F1 as feature values are lost at random. The tree ensembles degrade gracefully: with 30% of values lost, F1 is 85.2% for RF and 83.5% for XGBoost, against 79.1% for the MLP and 63.1% for LR. At 50% loss RF still reaches 81.3%. Bagging over many trees that split on different features appears to buffer the loss of any single value, whereas the linear model and, to a lesser extent, the MLP combine all features at once.

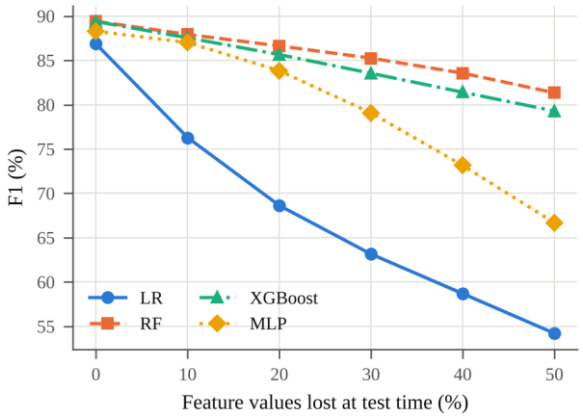


Figure 5: F1 under random loss of feature values at test time (mean of three repetitions; bands show one standard deviation)

Module outages reveal a different pattern (Figure 6). Losing the Basic group, which contains the TTL features, costs LR 36.5 F1 points, RF 10.3, XGBoost 6.9 and the MLP only 3.3. Losing the Content group costs 2.1 to 9.5 points and the Time group 0.8 to 3.6, while the General-purpose and Connection groups are nearly redundant. Model rankings therefore depend on the failure mode: the MLP is the second most fragile under random loss but the most robust when a whole module fails. Robustness claims should specify the failure mode against which they were tested, and in practice the health of the basic flow-statistics extractor deserves the closest monitoring.

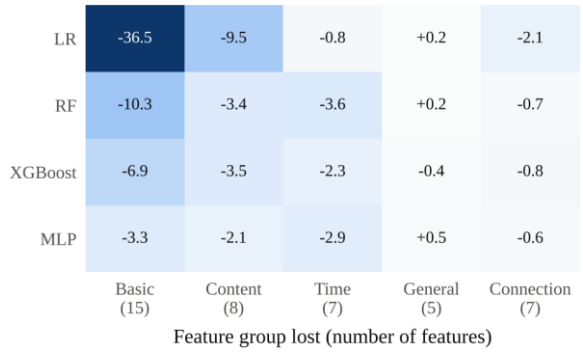


Figure 6: Change in F1 (percentage points) when an entire feature group is lost at test time; darker cells mark larger drops

5.7 Implications for deployment

Four practical implications follow. First, de-duplicate before training and report results on test flows that do not occur in training. Second, report detection at a stated false-alarm budget and check whether a threshold chosen on validation data holds on the test data. Third, test held-out attack families both with and without suspected artifact features, because generalization measured in the presence of a shortcut can be illusory. Fourth, test robustness against more than one failure model.

6. THREATS TO VALIDITY

The study uses a single dataset generated in a testbed, so the TTL shortcut is specific to UNSW-NB15 and the absolute results may not generalize to other networks; the evaluation method, however, is transferable. Leave-one-family-out is only a proxy for zero-day attacks, since families in one dataset share tools and infrastructure, which may make the proxy optimistic. Imputation by median or mode is one of several plausible failure models; a collector that drops whole flows or outputs stale values would behave differently. Hyperparameters were fixed rather than tuned, which avoids test-set leakage but may understate the best achievable scores. The main benchmark uses 3 seeds and shows small standard deviations, but the ablation, zero-day and degradation experiments use a single seed. Timing depends on hardware. Finally, the labels of the dataset were not independently verified.

7. CONCLUSION

On the official UNSW-NB15 partition, random forest and XGBoost achieve about 89% F1, but this figure hides a 27% false-alarm rate at the default threshold, thresholds whose false-positive rate grows by a factor of 2.4 to 4.9 between validation and test data, a detection rate for held-out attack families that falls from 90.1% to 75.1%, and a TTL shortcut that accounts for more than half of the apparent generalization to unseen Generic attacks. The detectors also differ substantially in their responses to telemetry loss, and their ranking depends on the failure mode. A detector intended for critical digital infrastructure should therefore be evaluated on duplication-free data, at a stated false-alarm budget, on held-out attack families with and without suspected artifacts, and under several failure models. Future work will apply this protocol across datasets using a common feature set [28], add temporally ordered splits, and study training with simulated missing features to improve robustness.

DATA AVAILABILITY

The UNSW-NB15 dataset is publicly available for academic research from its creators at https://research.unsw.edu.au/projects/unsw-nb15-dataset [31]; as its terms require, this paper cites the creators' publications [19, 20, 21, 22, 27].

8. REFERENCES

[1] G. Apruzzese, P. Laskov, and J. Schneider, "SoK: Pragmatic Assessment of Machine Learning for Network Intrusion Detection," in 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), 2023, pp. 592-614, doi: 10.1109/EuroSP57164.2023.00042.

[2] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and Don'ts of Machine Learning in Computer Security," in 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 3971-3988.

[3] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," ACM Transactions on Information and System Security, vol. 3, no. 3, pp. 186-205, 2000, doi: 10.1145/357830.357849.

[4] Bhavika and N. Duhan, "Intrusion Detection in the Era of Machine Learning: A Critical Survey of Algorithms and Evaluation Practices," International Journal of Computer Applications, vol. 187, no. 21, pp. 50-57, 2025, doi: 10.5120/ijca2025925297.

[5] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001, doi: 10.1023/A:1010933404324.

[6] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, 2016, doi: 10.1109/COMST.2015.2494502.

[7] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785-794, doi: 10.1145/2939672.2939785.

[8] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," BMC Genomics, vol. 21, no. 1, Art. no. 6, 2020, doi: 10.1186/s12864-019-6413-7.

[9] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," Neural Computation, vol. 10, no. 7, pp. 1895-1923, 1998, doi: 10.1162/089976698300017197.

[10] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study," in 2021 IEEE Security and Privacy Workshops (SPW), 2021, pp. 7-12, doi: 10.1109/SPW53761.2021.00009.

[11] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut

learning in deep neural networks," Nature Machine Intelligence, vol. 2, no. 11, pp. 665-673, 2020, doi: 10.1038/s42256-020-00257-z.

[12] S. M. Kasongo and Y. Sun, "Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset," Journal of Big Data, vol. 7, no. 1, Art. no. 105, 2020, doi: 10.1186/s40537-020-00379-6.

[13] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," ACM Transactions on Knowledge Discovery from Data, vol. 6, no. 4, Art. no. 15, 2012, doi: 10.1145/2382577.2382579.

[14] D. Kus, E. Wagner, J. Pennekamp, K. Wolsing, I. B. Fink, M. Dahlmanns, K. Wehrle, and M. Henze, "A False Sense of Security? Revisiting the State of Machine Learning-Based Industrial Intrusion Detection," in Proceedings of the 8th ACM on Cyber-Physical System Security Workshop (CPSS '22), 2022, pp. 73-84, doi: 10.1145/3494107.3522773.

[15] I. Linkov and A. Kott, "Fundamental Concepts of Cyber Resilience: Introduction and Overview," in Cyber Resilience of Systems and Networks, A. Kott and I. Linkov, Eds. Cham: Springer, 2019, pp. 1-25, doi: 10.1007/978-3-319-77492-3_1.

[16] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017.

[17] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," Nature Machine Intelligence, vol. 2, no. 1, pp. 56-67, 2020, doi: 10.1038/s42256-019-0138-9.

[18] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," Psychometrika, vol. 12, no. 2, pp. 153-157, 1947, doi: 10.1007/BF02295996.

[19] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in 2015 Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.

[20] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," Information Security Journal: A Global Perspective, vol. 25, no. 1-3, pp. 18-31, 2016, doi: 10.1080/19393555.2015.1125974.

[21] N. Moustafa, G. Creech, and J. Slay, "Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models," in Data Analytics and Decision Support for Cybersecurity, I. Palomares Carrascosa, H. K. Kalutarage, and Y. Huang, Eds. Cham: Springer, 2017, pp. 127-156, doi: 10.1007/978-3-319-59439-2_5.

[22] N. Moustafa, J. Slay, and G. Creech, "Novel Geometric Area Analysis Technique for Anomaly Detection Using Trapezoidal Area Estimation on Large-Scale Networks," IEEE Transactions on Big Data, vol. 5, no. 4, pp. 481-494, 2019, doi: 10.1109/TBDATA.2017.2715166.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, no. 85, pp. 2825-2830, 2011.

[24] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time," in 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 729-746.

[25] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," Computers & Security, vol. 86, pp. 147-167, 2019, doi: 10.1016/j.cose.2019.06.005.

[26] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid, "Developing Cyber-Resilient Systems: A Systems Security Engineering Approach," NIST Special Publication 800-160, Vol. 2, Rev. 1, National Institute of Standards and Technology, 2021, doi: 10.6028/NIST.SP.800-160v2r1.

[27] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems," in Big Data Technologies and Applications (BDTA 2020, WiCON 2020), Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 371. Cham: Springer, 2021, pp. 117-135, doi: 10.1007/978-3-030-72802-1_9.

[28] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a Standard Feature Set for Network Intrusion Detection System Datasets," Mobile Networks and Applications, vol. 27, no. 1, pp. 357-370, 2022, doi: 10.1007/s11036-021-01843-0.

[29] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in 2010 IEEE Symposium on Security and Privacy, 2010, pp. 305-316, doi: 10.1109/SP.2010.25.

[30] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.

[31] UNSW Canberra, "The UNSW-NB15 Dataset," [Online]. Available: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed Oct. 9, 2026).

[32] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," IEEE Access, vol. 7, pp. 41525-41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[33] N. B. Sharif, A. A. Susom, B. D. Sarker, S. Badjie and S. A. Ashraf, "Explainable Ethereum Fraud Detection for Secure Sustainable Digital Infrastructure," 2026 6th International Conference on Soft Computing for Security Applications (ICSCSA), Salem, India, 2026, pp. 539-544, doi: 10.1109/ICSCSA69367.2026.11708292.

[34] S. Badjie, A. A. Susom, B. D. Sarker, N. B. Sharif and S. A. Ashraf, "Ensembles or Graph Neural Networks? Detecting Illicit Bitcoin Transactions Under Temporal Drift," *2026 6th International Conference on Soft Computing for*

Security Applications (ICSCSA), Salem, India, 2026, pp. 557-561, doi: 10.1109/ICSCSA69367.2026.11707854.

[35] A. A. Susom, M. F. Ahammed, M. R. Labu, B. D. Sarker, S. Badjie and S. A. Ashraf, "FORTIS: A Hybrid Ensemble Model for Industrial Cybersecurity and Intrusion Detection," 2026 6th International Conference on Soft Computing for Security Applications (ICSCSA), Salem, India, 2026, pp. 1590-1596, doi: 10.1109/ICSCSA69367.2026.11708544.

[36] B. D. Sarker, M. F. Ahammed, M. R. Labu, S. Badjie, A. A. Susom and R. Hasan, "SecureEnsembleNet: Intelligent Cyber Threat Detection with Ensemble Learning," 2026 7th International Conference On Computational Vision and Bio Inspired Computing (ICCVBIC), Lalitpur, Nepal, 2026, pp. 950-955, doi: 10.1109/ICCVBIC71195.2026.11689560.

[37] M. Das, A. A. Susom, M. A. Bhuiyan, A. Al Maimun and R. Hasan, "Deep Learning for Cancer Type Identification from RNA-Seq Gene Expression: A Comparison with ML," *2026 International Conference on Modern Sustainable Systems (CMSS)*, Shah Alam, Malaysia, 2026, pp. 1404-1407, doi: 10.1109/CMSS69636.2026.11688231.

[38] D. Nandy, M. O. Bouraima, S. P. Koneru, M. B. N. Shahin, A. Biswas and M. J. Islam, "Resilient Digital Twin Estimation for Smart Manufacturing Systems under Sensor Data Loss," *2026 9th International Conference on Inventive Computation Technologies (ICICT)*, Kirtipur, Nepal, 2026, pp. 264-270, doi: 10.1109/ICICT68280.2026.11511027.

[39] M. O. Bouraima, M. B. N. Shahin, D. Nandy, M. J. Islam, S. P. Koneru and A. Biswas, "Trustworthy AI for Predictive Analytics under Missing Data," *2026 9th International Conference on Inventive Computation Technologies (ICICT)*, Kirtipur, Nepal, 2026, pp. 2141-2148, doi: 10.1109/ICICT68280.2026.11510904.

[40] U. Ahmed, M. R. Chowdhury, M. S. Mia, A. Chowdhury, N. Bakchi and M. I. Hossain, "Leakage-Aware Benchmarking for Trustworthy Supply Chain Risk Prediction," *2026 10th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Kirtipur, Nepal, 2026, pp. 1137-1143, doi: 10.1109/I-SMAC70384.2026.11747631.

[41] J. Shekh, D. Nandy, M. B. Nuruzzaman Shahin, M. Jakaria Islam, A. Biswas and M. F. Ahmed, "Explainable AI and SHAP-based Modeling for Workplace Risk and Safety Analytics," 2026 International Conference on Computer Networks and Inventive Communication Technologies (ICCNCT), Erode, India, 2026, pp. 1774-1781, doi: 10.1109/ICCNCT68477.2026.11590467.

[42] Hossain, M., & Uddin, M. B. (2025). Digital twins and federated learning for industrial internet of things. *Int. J. Sci. Res. Arch*, *16*(1), 729-736.

[43] M. B. Uddin, M. Hossain, T. Mahmud and S. Das, "Blockchain Integration in IoMT for Secure Healthcare: Challenges, Integration, and Solutions," *2025 6th International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2025, pp. 597-604, doi: 10.1109/ICIRCA65293.2025.11089683.

[44] S. P. Koneru, M. Omanda Bouraima, R. Rahman and M. Hossain, "Object Detection in Unstructured Driving Environments Using NASNetMobile and InceptionV3," *2025 9th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 2025, pp. 1945-1949, doi: 10.1109/ICECA66444.2025.11382875.

[45] Md Hossain & Nakul Bakchi & Md Imran Hossain & Suman Das & Md Faysal Ahmed, 2026. "Blockchain-Driven Federated Learning for Secure Industrial Predictive Maintenance," International Journal of Innovative Science and Research Technology (IJISRT), vol. 11(06), pp. 2442-2448.